UCRL-ID-148122

LLNL I/O Testbed Analysis of IBM M80 (P660 6MI)

K. Fitzgerald, J. Daveler, T. Heer, M. Gleicher

February 1, 2002

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

LLNL I/O Testbed Analysis of
IBM M80 (P660 6M1)

Keith Fitzgerald
Jim Daveler
Todd Heer
Mike Gleicher

February 2002

Index

Background: 3
Goal: 3
Target configuration: 3
Test Plan including Test results: 8
Diagnostic simulation of HPSS behavior: (Keith Fitzgerald) 8
HPSS Based Testing (Jim Daveler) 32
HSI data rates (Mike Gleicher) 37
SP Attached M80 Analysis (Todd Heer) 42
Conclusions 44
Appendix A: Network options 46
Appendix B: Outstanding Issues 48
Appendix C: Individual Card performance (reference) 49

Background:

The HPSS hierarchical storage system currently uses “mover” nodes to interface client processes to the hardware devices. As archival bandwidth requirements increase, the number of mover nodes required to provide the necessary bandwidth has become a significant factor both in terms of capital expense and administrative overhead. The LLNL data storage group would therefore like to select the most powerful yet cost effective IBM architecture currently available for use as future HPSS movers nodes. The LLNL storage group currently uses IBM “winterhawk2” SP nodes as HPSS mover nodes. Several IBM architectures including “nighthawk” SP nodes and several “P” series IBM architectures were considered and the “M80” system appeared to be the most promising based upon its high memory bandwidth and capability to attach multiple “RIO” drawers offering 64bit 66MHz PCI buses.

Goal:

The goal of this testbed analysis is to determine the usefulness of the IBM M80 as an HPSS mover and to identify I/O bottlenecks in this architecture.

Target configuration:

The IBM 7026-M80 being tested is really an *obsolete* machine. It was replaced by the pSeries 660 Model 6M1 on 9/4/01. However the hardware differences between the M80 and the P660-6M1 are all located in the processor used – NOT in the memory or I/O areas. In fact, current IBM M80 customers can upgrade their systems to the new model architecture by replacing their current processor cards much the way we upgraded from nighthawk1 to nighthawk2 systems. Therefore since our goal is to saturate the I/O and/or memory capability of the machine and the new 6M1 systems are not available for borrows, we believe our I/O test results will be valid and we can extrapolate results in the processor area. The following diagrams were found in the “IBM eserver pseries 660 model 6M1 Technical Overview and Introduction. Written by Stephen Lutz and Shyam Manohar September 4,2001.

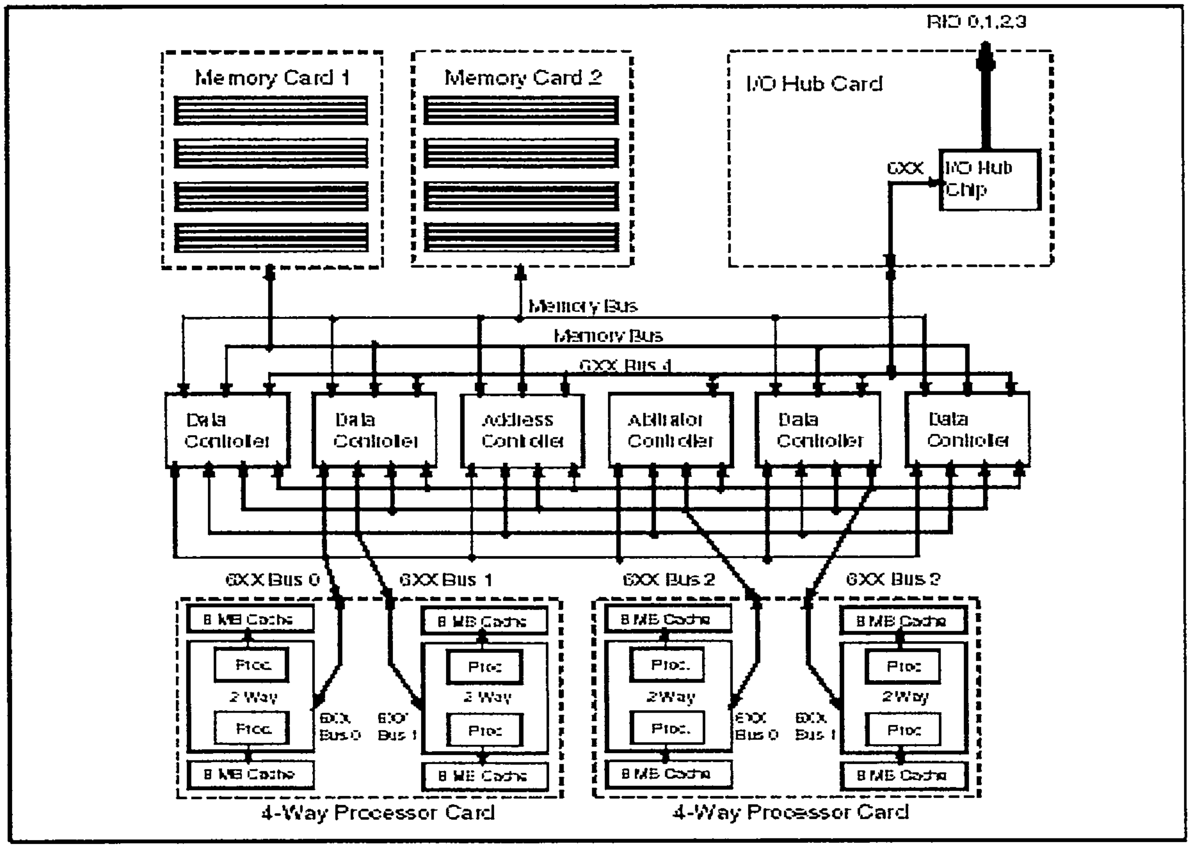


Figure 2. RS/6000 Model 6M1 System Schematic within the CEC

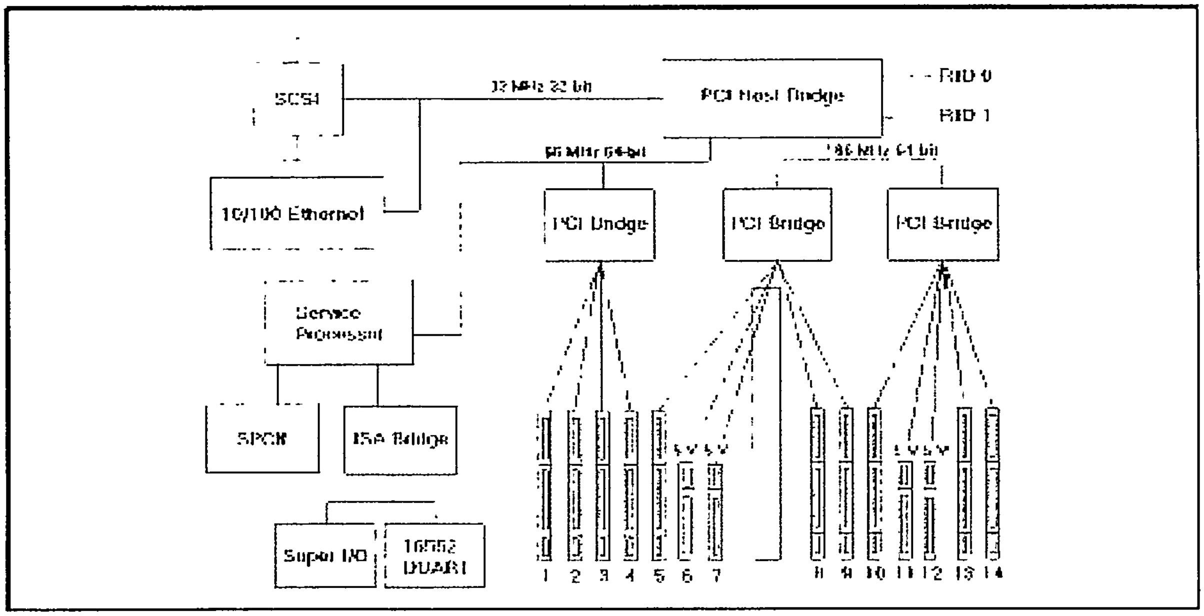


Figure 5. Slot Layout of the I/O Drawer

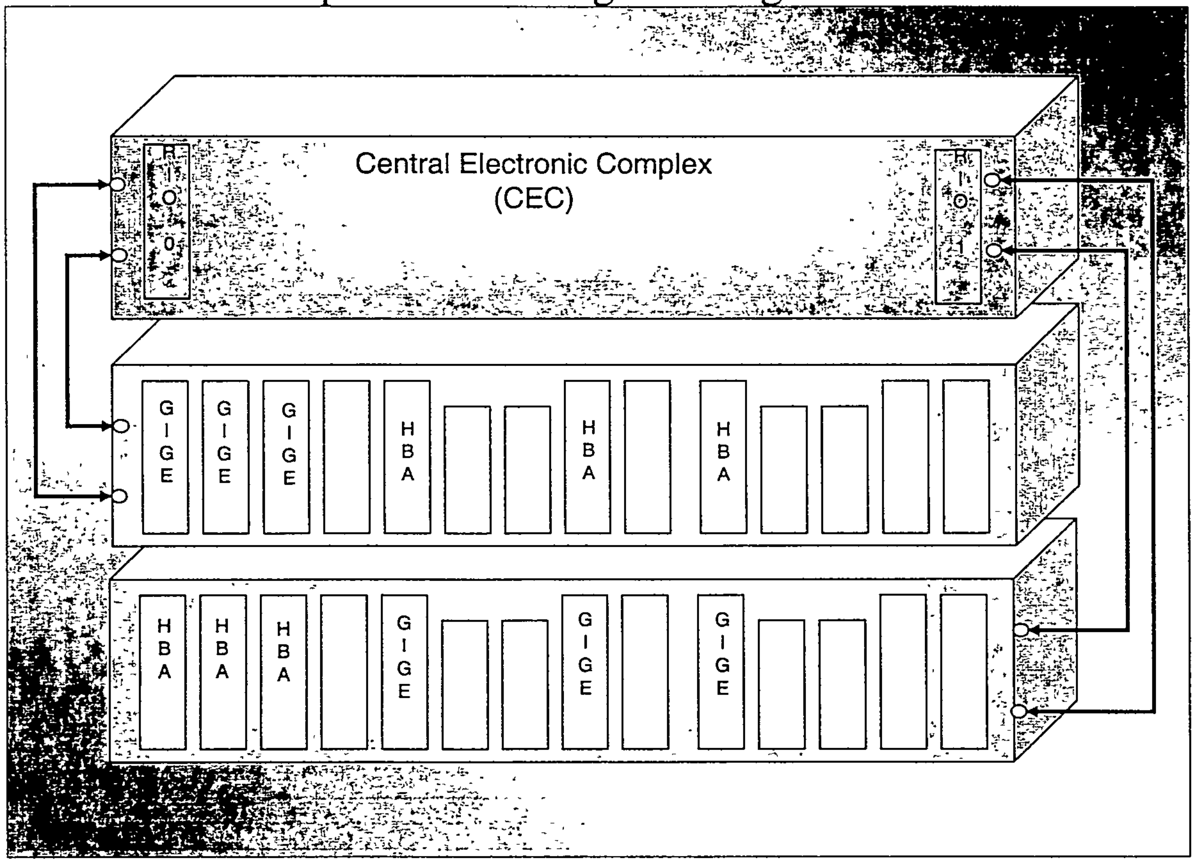
Bus Bandwidth

The following are the theoretical maximum bandwidths, as applicable for an 8-way 750 MHz SMP configurations:

- Bandwidth of the bus between each memory riser card to memory controller: 4.8 GB/s
- Bandwidth of the PowerPC 6xx bus used to interface each pair of processors: 2.4 GB/s
- Bandwidth of the PowerPC 6xx bus used to interface the I/O hub: 2.4 GB/s
- Aggregate memory bandwidth: 9.6 GB/s
- Aggregate processor bandwidth: 9.6 GB/s
- Four drawer I/O bandwidth: 4 GB/s (4 x 500 MB/s bi-directional)

Analysis of the system architecture leads us to believe that the worst restriction in the M80 is the 2GB/s 6XX bus that connects the I/O hub to the memory controllers (16 bytes@125Mhz =2GB/s). If we assume the same bus utilization that we achieved on the winterhawk nodes (300MB/s of a possible 480MB/s = 63%) that means we can expect 1,260MB/s on the M80 if this is indeed the choke point. 1260MB/s translate to a quantity of roughly 12 I/O interfaces each running at 100MB/s. Each RIO loop can run at 1GB/s in each direction (500MB/s * 2 connections) – therefore two RIO drawers should give us 2 GB/s (theoretical) in each direction – more than enough to saturate the 6XX bus. Within each RIO drawer you have two 64bit 66Mhz PCI buses. Each PCI bus should be capable of running at 528MB/s and should actually be able to achieve around 60% of the theoretical rate or 316MB/s. If we install three 100MB/s I/O cards in each PCI bus and use two RIO drawers, we should have a configuration which can saturate the 6XX I/O bus leading from the I/O Hub card to memory while transferring data at full speed in either direction (12cards * 100MB/s each). The proposed target configuration follows. Due to the full duplex nature of the cards, the machine is actually oversubscribed by a factor of 2. Note that we installed like cards in each PCI bus so a device with a high interrupt priority should not be able to starve the other cards on the PCI bus.

Proposed M80 Target Configuration



A series of tests will be run which will hopefully verify the proposed configuration. The goal of the testing will be to find the point at which the I/O capability of the hardware has been exhausted. However before beginning the I/O stress tests we will attempt to verify that each I/O device, PCI bus, and RIO drawer performs as expected when tested individually. Once baseline performance has been verified, we will run a series of tests designed to incrementally increase the I/O load on the test configuration until the throughput peaks. Past experience has shown us that I/O throughput dramatically degrades when the system saturates. Once we have determined the optimal I/O load for the test configuration we will attempt to determine the number of processors and memory required to support the I/O load. We will then configure the test configuration as an HPSS mover in order to determine how closely our test results match actual system behavior. As a result of the testing will be an optimized mover configuration, which will deliver a known I/O performance. Since our test configuration will hopefully support (at least) six gige interfaces we will have to configure two additional jumbo gige subnets in the I/O testbed.

IBM Hardware:

IBM M80 with 8 processors, 32 GB memory and 3 additional RIO drawers (4 total).

LLNL hardware:

6 IBM gigabit Ethernet cards
6 emulex LP8000 Host Bus Adapters
Brocade 4400 16-port fibre switch
2 TB Data Direct Networks (DDN) fibre RAID disk. (4 tiers capable of running at 96MB/s on each interfaces)
1 TB MetaStor fibre RAID disk (2 interfaces capable of running at 80MB/s each)
1 IBM nighthawk2 16 processor 8GB memory SP node with two RIO drawers (two gigabit interfaces in each)
4 IBM winterhawk2 4 processors 2GB memory SP nodes each with one jumbo gigabit Ethernet interface.

Test Plan including Test results:

(see actual test log for more specific information).

Diagnostic simulation of HPSS behavior: (Keith Fitzgerald)

1. Install and update operating system patch level.

   AIX version 4.3.3 was installed and updated to level 50.

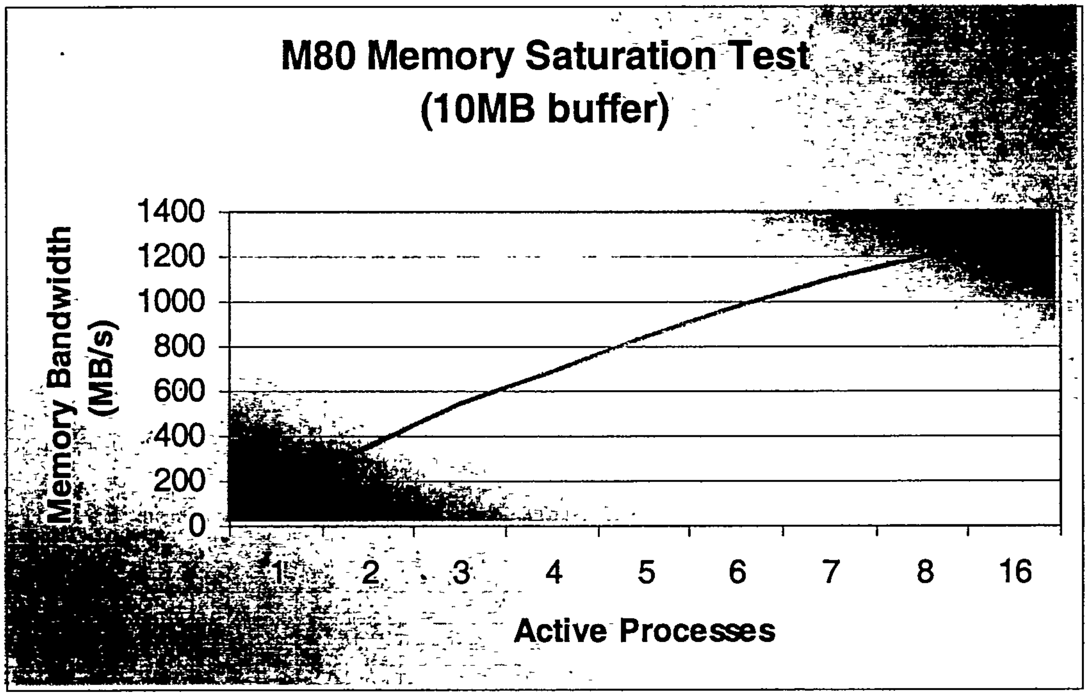
```
  </u/keith>lslpp -l | grep bos
      bos.rte.bosinst   4.3.3.50  COMMITTED  Base OS Install
```

2. Test Memory bandwidth

   Historically, we've found a high correlation between memory bandwidth and performance as an HPSS mover. The "Memtest" program exercises memory by allocating two buffers and copying data between them. You increase the number of concurrent processes until the machine's memory bandwidth is saturated. We have data on silver nodes, winterhawk nodes, and nighthawk nodes which can be compared to the M80 performance.

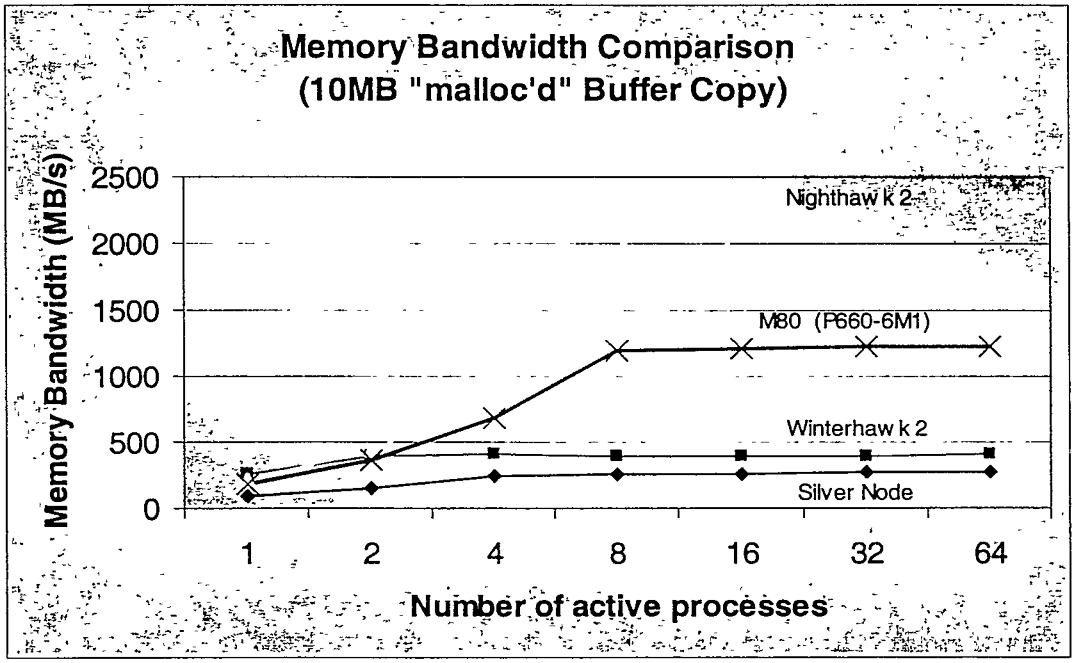
Tested memory using "memtest" 10MB buffer 100 copies

Processors	1	2	3	4	5	6	7	8	16
Bandwidth (MB/s)	186	369	548	689	840	977	1098	1202	1216



Note that there's enough memory bandwidth for all processors (meaning that the data rate increases as each process is added, asymptotically approaching the memory saturation value). Aggregate bandwidth increases as each physical processor becomes active.

The following graph compares M80 memory bandwidth to other architectures we have tested. A 16 processor Nighthawk2, a 4 processor Winterhawk2, and a 4 processor Silver Node.



Memory Bandwidth Comparison
(10MB "malloc'd" Buffer Copy)
Memory Bandwidth (MB/s)
2500
2000
1500
1000
500
0
Nighthawk 2
M80 (P660-6M1)
Winterhawk 2
Silver Node
1
2
4
8
16
32
64
Number of active processes

3. Installed four IBM gigabit Ethernet interfaces in the first PCI bus (slots 1,2,3,4) of the RIO drawer daisy chained to the primary RIO drawer.

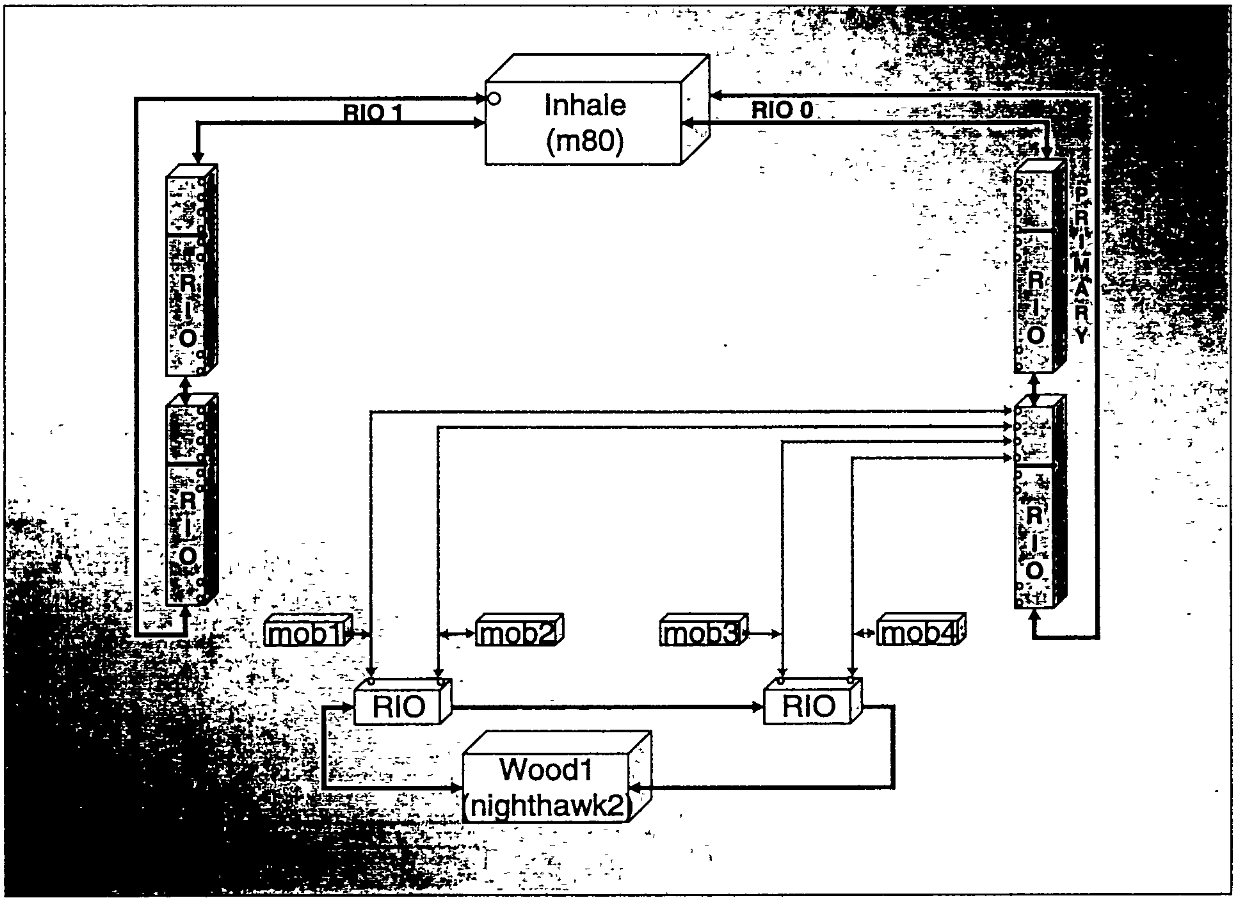
```
Adjusted the following network options: (values from mob1 an HPSS
  production mover).. other default values unmodified
Changes made at boot time by /etc/rc.net

           option            was     new
          thewall          65536   1048496
            sbmax        2097152   left it ... larger (1048576)
           ifsize              8        35
        ncb_limit              0   left it ....
    ncb_pseg_limit      16777192   left it .. larger (1048496)
      ipforwarding             0   left it ...
     tcp_sendspace         16384    655360
     tcp_recvspace         16384    655360
     udp_sendspace         65536    655360
     udp_recvspace        655360    655360 ... same
    nonlocsrcroute             0         1
        tcp_mssdflt         9000   left it  ..  (1448 on mob1)
          rfc1323              0         1
 udp_pmtu_discover             0         1
 tcp_pmtu_discover             0         1
         ipqmaxlen           100       128
    ipsrcrouterecv             0   left it      (1 on mob1)

 two new parms:
      tcp_newreno              0
   tcp_nagle_limit         65535

See appendix A for complete network options (no).

Configured the interfaces as follows: (2-5)
chdev -l ent2 -a jumbo_frames=yes -a tx_que_size=2048 -a \
  rx_checksum=yes
chdev -l en2 -a netaddr=134.9.33.134 -a netmask=255.255.255.240 \
  -a state=up -a mtu=9000 -a rfc1323=1 -a tcp_sendspace=1048576  \
  -a tcp_recvspace=1048576
    (same procedure for other jumbo gige interfaces)

Brought the four network interfaces up (one on each jumbo subnet)
Installed netperf and netserver (copied from police to
  /usr/local/bin)
```

The following series of tests will be performed using diagnostic routines (netperf for gige cards and a simple "home grown" memory based disk test which selectively reads or writes large blocks of disk data).

LLNL runs four gigabit Ethernet subnets in our production environment. The I/O testbed is configured to mirror our production environment as much as possible. In the following diagram, the M80 (inhale) has four gigabit interfaces, one configured on each of the four jumbo subnets. Our test environment also included a nighthawk2 node (wood1) which also supports four jumbo gigabit Ethernet interfaces. The four winterhawk2 nodes (mob1-4) each have a single jumbo gigabit Ethernet interface and are configured one on each of the four jumbo Ethernet subnets. On inhale, en2-5 were the jumbo subnets.

Jumbo subnet	M80 interface	Nighthawk	Winterhawk
1	inhale-en2	wood1-en2	mob1-en2
2	inhale-en3	wood1-en3	mob2-en2
3	inhale-en4	wood1-en4	mob3-en2
4	inhale-en5	wood1-en5	mob4-en2

4. Verify individual data rates of each M80 gige adapter. We used the first PCI bus in the RIO drawer daisy chained to the primary RIO. This test used the winterhawk nodes as data source or sink because they are completely independent.

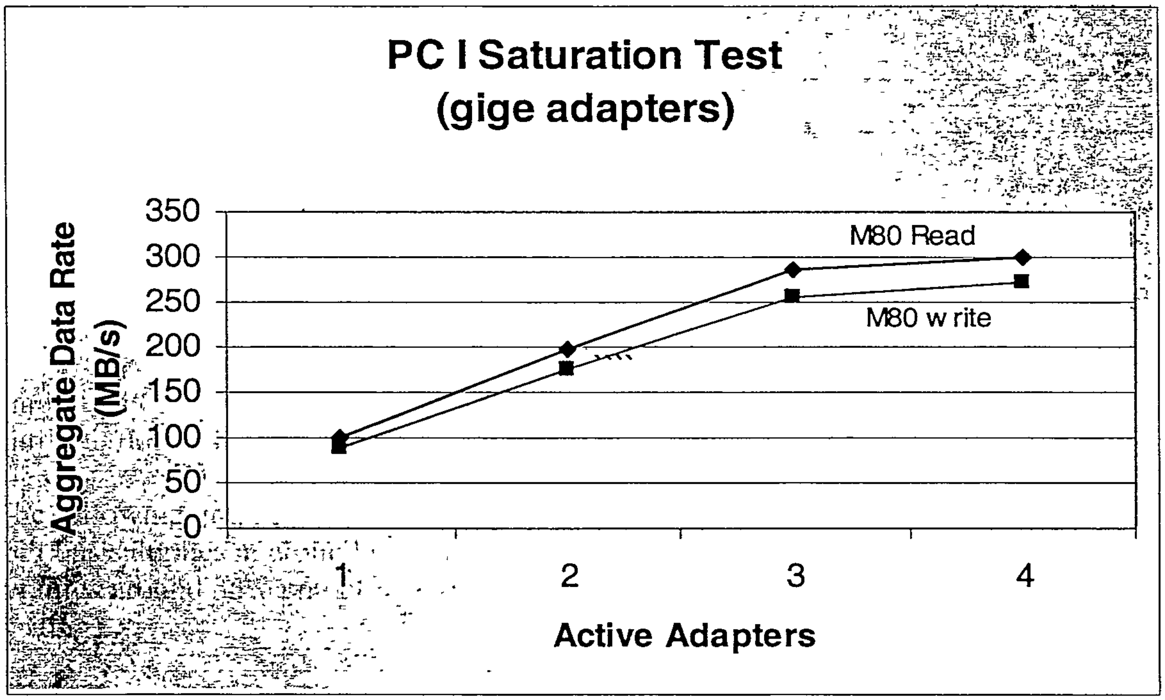


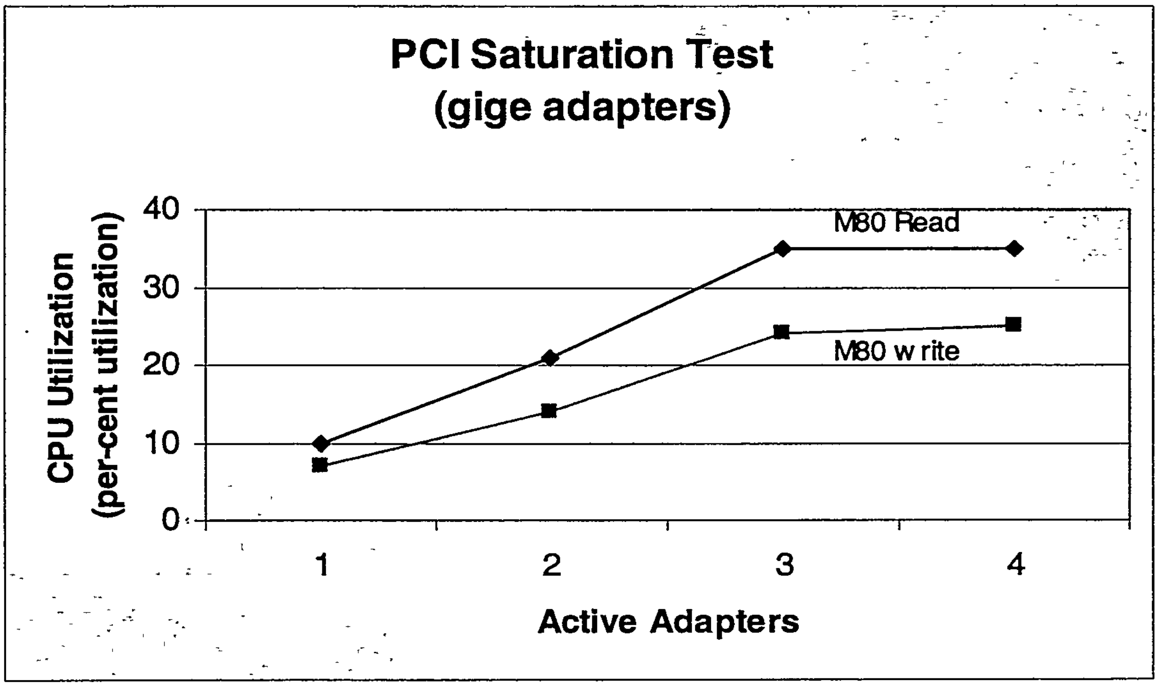
Source	Destination	Data rate (MB/s)	CPU Utilization (M80 perspective)
inhale-en2	mob1-en2	88	7%
inhale-en3	mob2-en2	88	7%
inhale-en4	mob3-en2	88	7%
inhale-en5	mob4-en2	85	7%
mob1-en2	inhale-en2	100	9%
mob2-en2	inhale-en3	100	9%
mob3-en2	inhale-en4	100	9%
mob4-en2	inhale-en5	100	9%

Note that at this point I found that the jumbo gige cards in the four winterhawk2 nodes (mob1-mob4) were located in the 32bit 33MHz PCI bus. This impacted the peak and full duplex performance of the gige cards installed in the M80. The winterhawks were reconfigured and the jumbo interface moved to one of the 64bit 33MHz PCI buses later in the testing.

5. Saturate PCI bus by activating gige cards one at a time and re-testing performance until full device bandwidth can not be achieved

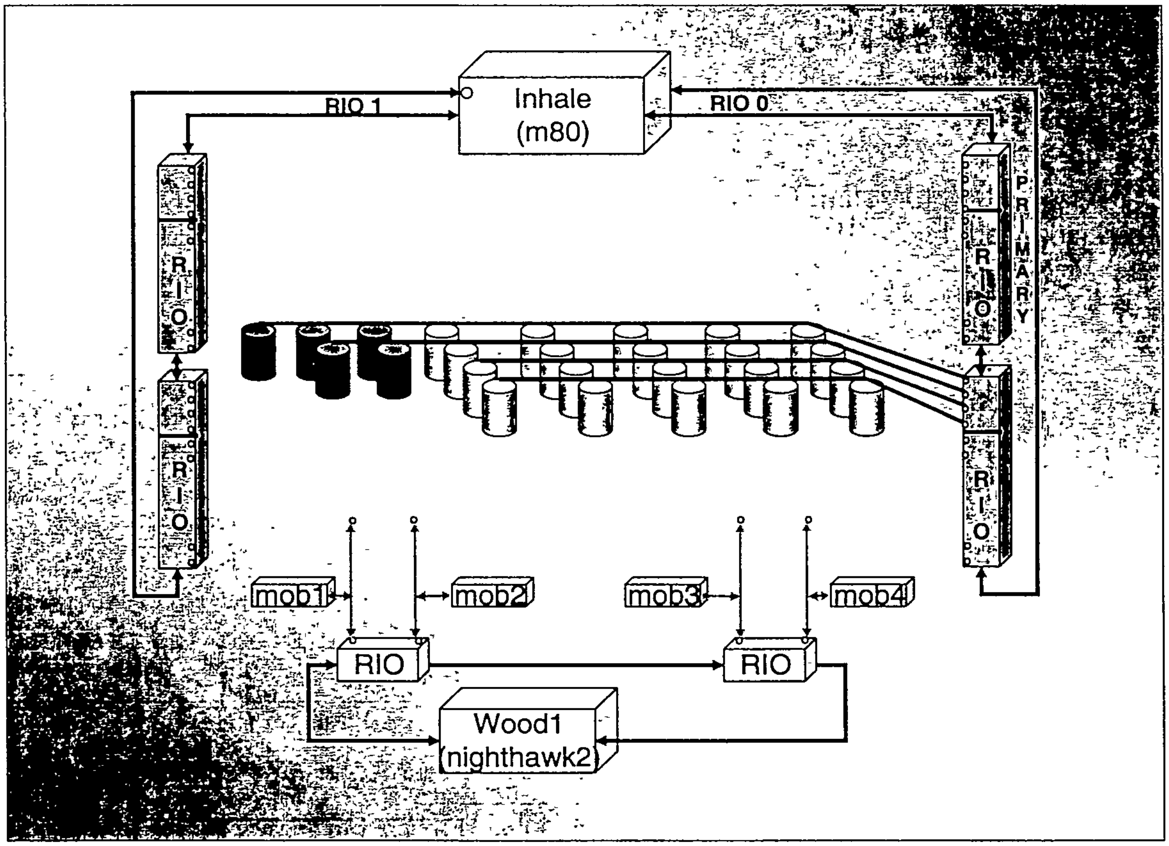
   The following graphs show that using IBM gigabit Ethernet adapters a 64bit 66MHz PCI bus saturates at around three concurrent transfers. Maintaining three transfers requires around 35% cpu utilization (35% of 8 cpu's) for a read or 25% when writing.




PCI Saturation Test
(gige adapters)
CPU Utilization
(per-cent utilization)
40
30
20
10
0
1
2
3
4
Active Adapters
M80 Read
M80 w rite

6. Remove the four gige adapters and replace them with four emulex LP8000 Host Bus Adapters (HBA's). Verify the performance of each emulex HBA and the associated disk device. (same goal as step 4 above).

```
Installed emulex version 4.1.0.5 fibre channel driver.
Upgraded emulex HBA microcode to d382a1.awc (kernel
 revision LP8K 2.03x15) (initially got bad rates on 3
 down level adapters!)
changed the following default parameters for fcosdisk:
    attribute=mode_data    default d= ""    (was a hex #)
    attribute=queue_depth  default = 64     (was 8)
changed the following default parameter in lp8000
    attribute=lun_queue_depth default = 64 (was 30)
```

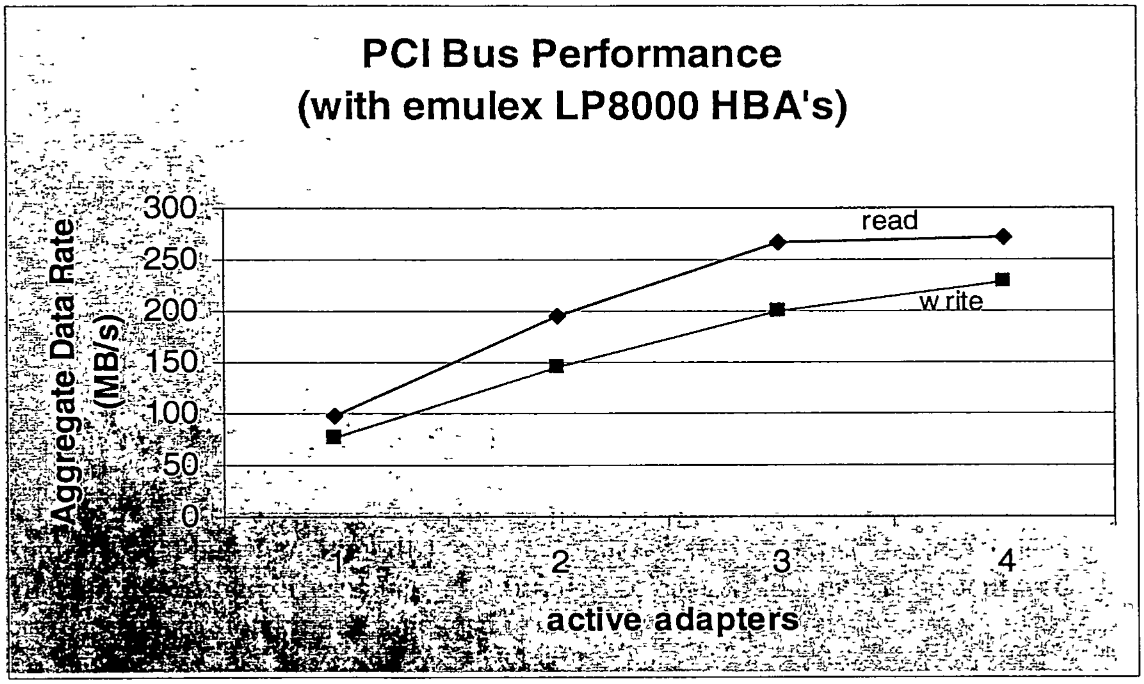


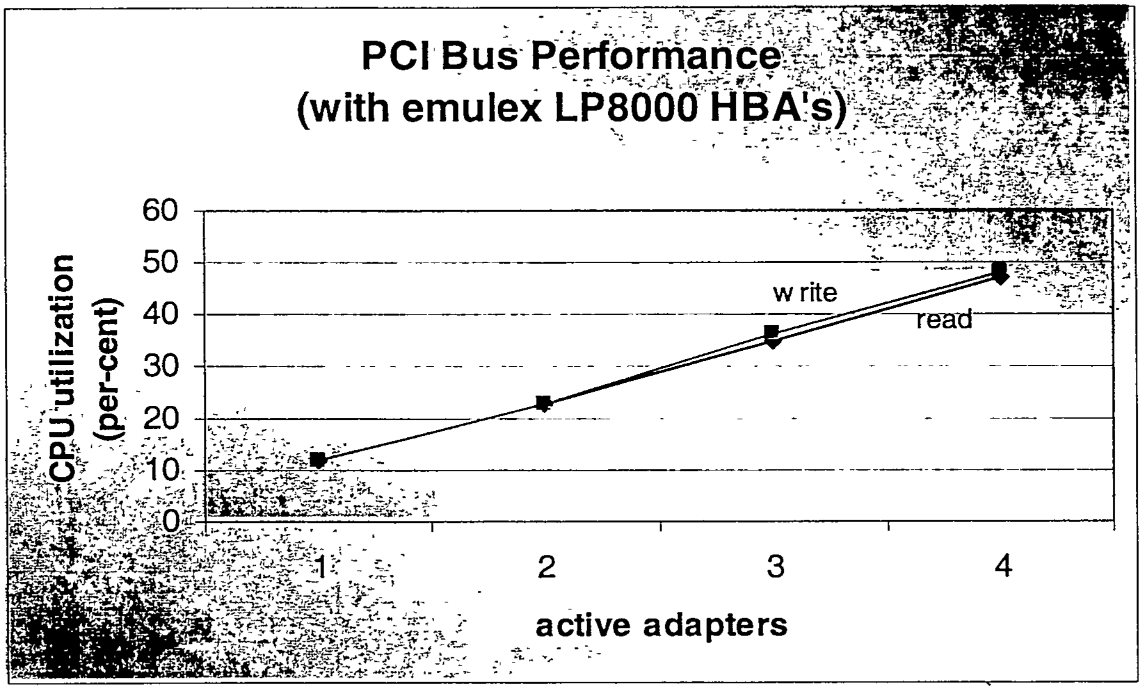
I utilized the M80 "hot swap" capability to remove the gige adapters and install the HBA's . It worked great and the M80 RIO drawer design is a big improvement over the nighthawk RIO. In the actual test configuration, the M80's HBA's were connected to a 16 port Brocade 2801 fibre switch. All disk devices were also attached to the switch. This allowed me to verify disk behavior from the winterhawk2 nodes without any recabling or reconfigurations. Although multiple logical units (LUN's) were configured on each physical RAID device, care was taken during testing to assure that no more than one data stream was active to each physical device. Our network attached disk test environment includes four "tiers" (physical RAID devices) of Data Direct Networks (DDN) disk , each with a dedicated fibre interface. The DDN disks are capable of reading and writing at 96MB/s based upon tests using our IBM winterhawk2 nodes and emulex LP8000 HBA's via the Brocade switch. We also have a "metastor" fibre disk device with two fibre interfaces. The "metastor" disk is capable of reading and writing at about 80MB/s, again verified using our IBM winterhawk2 nodes,using emulex LP8000 adapters and the brocade switch.

	READ (MB/s)	WRITE (MB/s)	READ/WRITE (MB/s)
Winterhawk data rates (baseline)	98	98	150-160
M80 data rates	98	77	112

These tests used an 8MB read/write size in the test program. M80 HBA write rates are VERY low! We get 98 from our winterhawk2 nodes. We are currently working with emulex to resolve the problem. During these individual HBA tests, I also discovered that the LP8000's full duplex performance is also bad on the M80. We later installed and tested an emulex LP9000 and saw very similar performance.

7. Saturate PCI bus by adding disk streams (unique HBA and disk device) one at a time and re-testing performance until full device bandwidth can not be achieved. (corresponds to step4 in the gige testing).

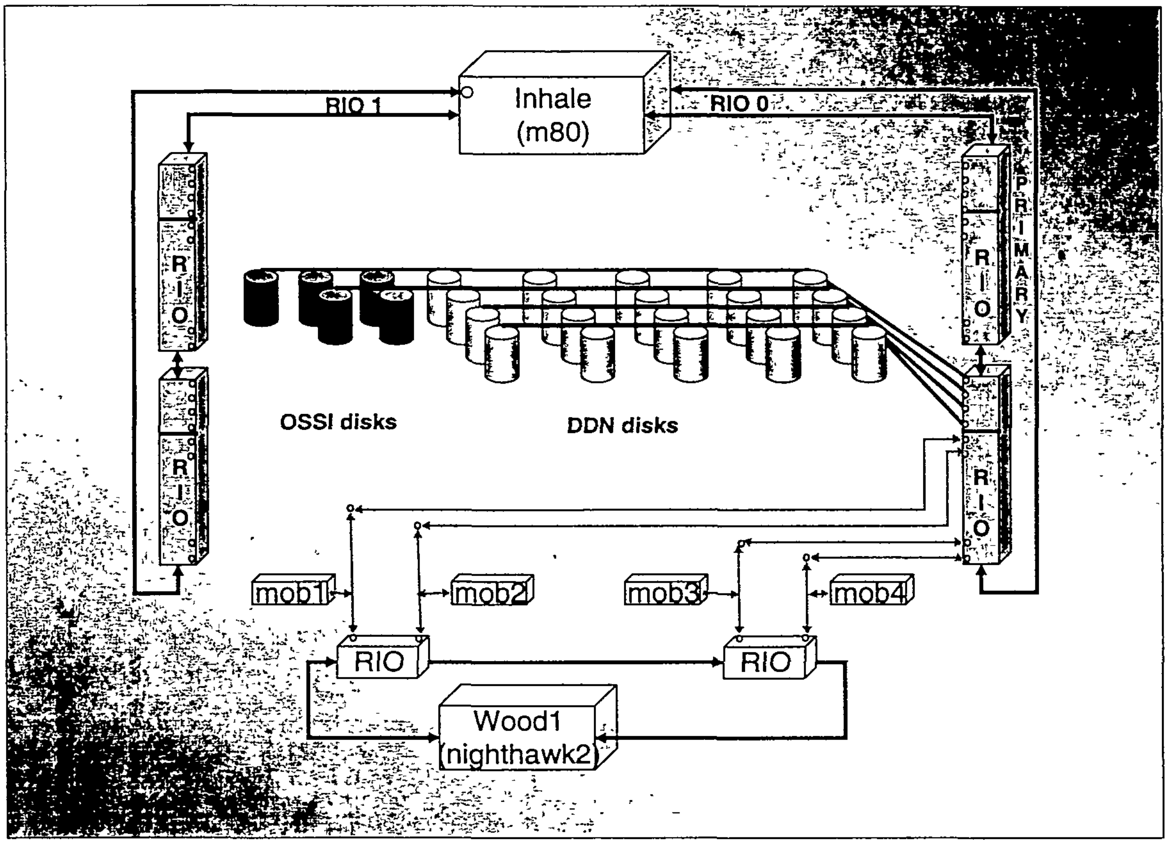




This test also showed that three adapters can operate at (or near) full speed concurrently running half-duplex. A fourth adapter saturates the PCI bus. I also attempted to saturate the PCI bus with a combination of reads and writes.
I also ran a test that wrote to all four DDN disks while concurrently reading the metastor disks. Two HBA's were reading AND writing concurrently. Two HBA's were just writing. I achieved a total of 89MB/s on the two reads and a total of 161MB/s on the four writes for a total of 250MB/s and 68% CPU utilization. Note that the CPU utilization (wait time) increased even tho the aggregate bandwidth did not.

Adapter	Read (MB/s)	Write (MB/s)	Total (MB/s)
lpfc0	44	31	75
lpfc1	45	30	75
lpfc2	-	51	51
lpfc3	-	49	49
			250

8. Verify that both buses in an individual RIO drawer can run at full speed concurrently.



I installed the four IBM gige cards tested in step 3 in the 2nd PCI bus of the RIO drawer containing the four LP8000 HBA's. After verifying the individual performance of the gige cards in their new location, I attempted to run all four gige cards concurrently. I found that one of the cards performed at full speed and the other three cards split the remaining bandwidth. I believe that this is caused by the PCI bus repeaters. The bridge repeaters seem to distribute the bandwidth available on the primary bus evenly to the two secondary buses. I achieved around 325MB/s total reading from all four interfaces concurrently.

Interface	3 active interfaces	4 active interfaces
en2 (61-08)	98	74
en3 (6A-08)	99	74
en4 (6D-08	99	74
en5 (71-08)	-	102
	296	324

This is an interesting discovery but not the relevant to the test. It is, however the highest aggregate PCI bus data rate so far....
Theoretical PCI bus rate=8bytes/cycle * 66M cycles/sec=528MB/s
PCI bus saturation=324/528=61% theoretical

Now that the gige performance had been verified, I setup a test that read four DDN disks on the first bus while concurrently reading four gige adapters on the second PCI bus. Got a total of 375MB/s … with 25% CPU utilization.

RIO Saturation READ test

PCI bus	Adapter	Device	Rate
1	lpfc0	DDN-tier1	41
1	lpfc1	DDN-tier2	42
1	lpfc2	DDN-tier3	43
1	lpfc3	DDN-tier4	43
2	en2	mob1-en2	41
2	en3	mob2-en2	41
2	en4	mob3-en2	41
2	en5	mob4-en2	83
			375MB/s

Then I tried the same thing with writes ….. 325MB/s with 17% CPU utilization.

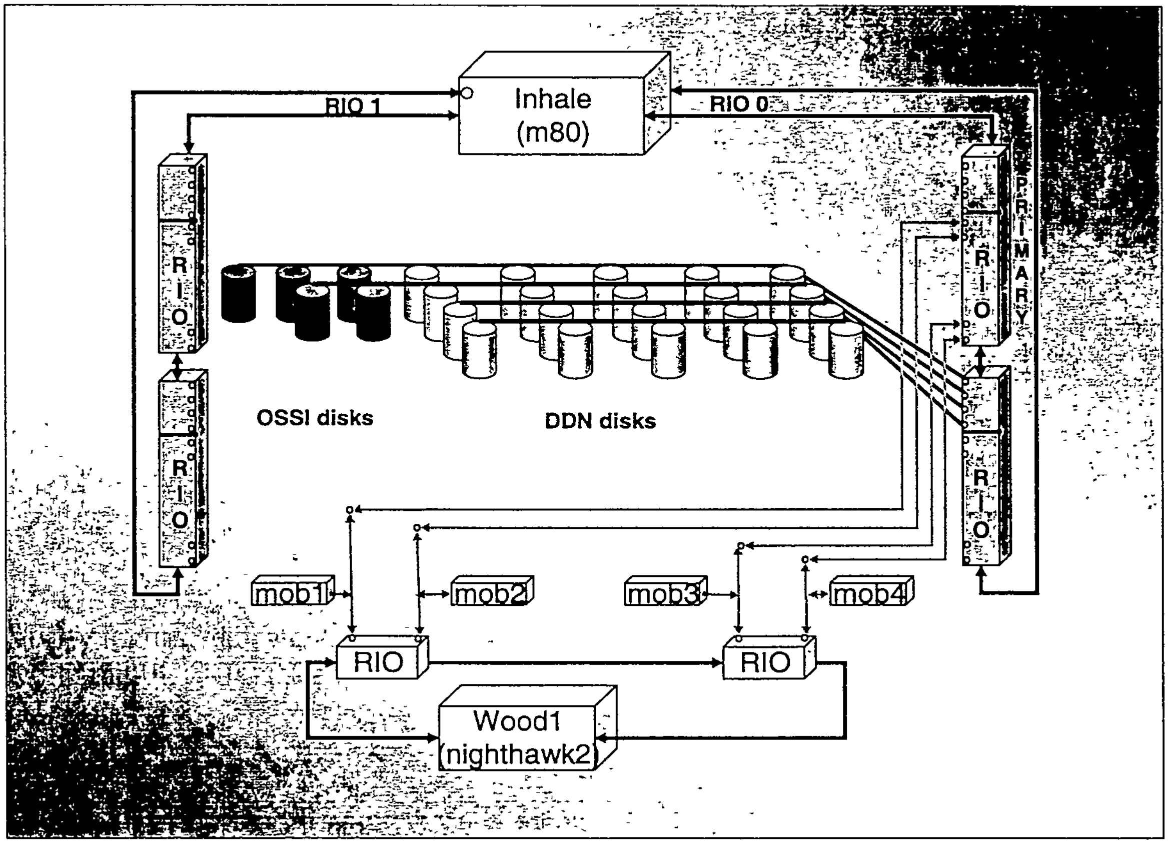
RIO Saturation WRITE test

PCI bus	Adapter	Device	Rate
1	lpfc0	DDN-tier1	40
1	lpfc1	DDN-tier2	38
1	lpfc2	DDN-tier3	40
1	lpfc3	DDN-tier4	40
2	en2	mob1-en2	40
2	en3	mob2-en2	40
2	en4	mob3-en2	40
2	en5	mob4-en2	47
			325MB/s

THIS IS SIGNIFICANT …the rates achieved indicate that an individual RIO drawer can not make use of both RIO connections! The RIO daisy chain architecture seems to only add redundancy – you can't seem to make use of the bandwidth! I suspect that you will have to add the daisy-chained RIO drawer in order to make use of the second RIO connection to the CEC.

A RIO connection's theoretical data rate is 500MB/s.
RIO connection saturation seems to be 375/500=75% theoretical

9. Verify that the daisy chained RIO drawer can make use of the second RIO connection while the first RIO uses the first connection.



To test this theory I moved the four gige cards to the primary RIO drawer and reran the previous test sequence.
Got 554MB/s reading 8 devices with 44% cpu utilization.

RIO Saturation READ test

PCI bus	Adapter	Device	Rate
1	lpfc0	DDN-tier1	74
1	lpfc1	DDN-tier2	51
1	lpfc2	DDN-tier3	74
1	lpfc3	DDN-tier4	74
2	en2	mob1-en2	63
2	en3	mob2-en2	63
2	en4	mob3-en2	63
2	en5	mob4-en2	96
			554 MB/s

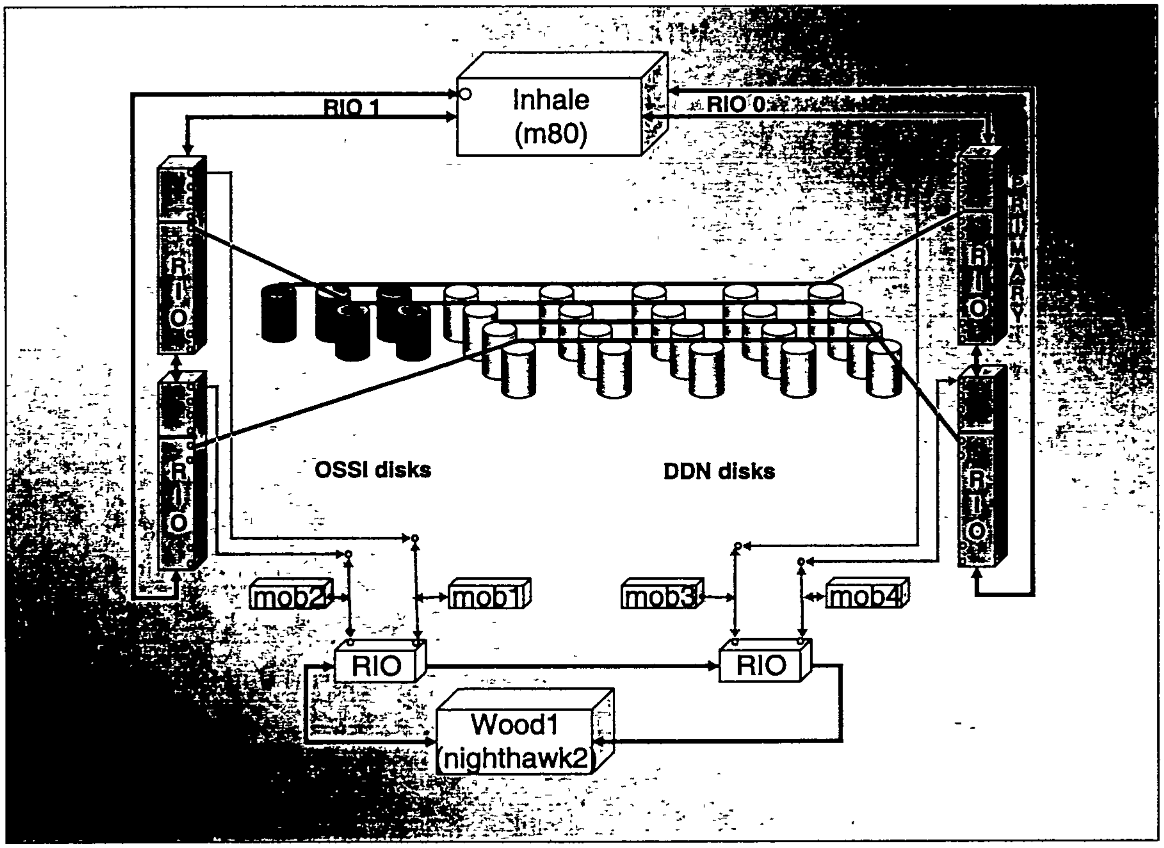
Then I tried the same thing with writes ..... 501MB/s with 28% CPU utilization.

RIO Saturation WRITE test

PCI bus	Adapter	Device	Rate
1	lpfc0	DDN-tier1	57
1	lpfc1	DDN-tier2	54
1	lpfc2	DDN-tier3	57
1	lpfc3	DDN-tier4	56
2	en2	mob1-en2	62
2	en3	mob2-en2	62
2	en4	mob3-en2	62
2	en5	mob4-en2	91
			501MB/s

Based upon the data rates observed we were indeed able to utilize the second RIO channel.

10. Saturate the I/O hub (or the 6XX bus connecting the I/O hub to memory).



At this point I assumed that the second pair of RIO channels and RIO drawers would exhibit the same behavior as the pair I just tested. I distributed the 8 cards between the four RIO drawers, one card per PCI bus. Based upon previous testing, each card should be able to run at full speed in this configuration. Because each card is capable of running full duplex, theoretically we should be able to produce a load of 8*200MB/s=1600MB/s which should saturate the 6XX bus. Actually, we know that the emulex HBA will only run at a bit over 1GB/s full duplex on the M80.

Test 1: read four DDN disks, along with four gige writes to winterhawk nodes. Got 731MB/s with 37% system time and 42% wait time.

Interface	read	write
lpfc0	97	-
lpfc1	97	-
lpfc2	97	-
lpfc3	97	-
en2	-	91
en3	-	91
en4	-	70
en5	-	91
	388	343 = 731

Test2: read four DDN disks, read four nighthawk gige interfaces, write four winterhawk node gige interfaces. Got 782MB/s 86% system time and 7% wait time.

Interface	read	write
lpfc0	92	-
lpfc1	90	-
lpfc2	92	-
lpfc3	92	-
en2	45	58
en3	47	58
en4	47	58
en5	45	58
	550	232 = 782

The disks ran pretty much as expected but the gige's were impacted. The same gige transfers performed without disk I/O performed at around 75MB/s each at 75% system time.

Test3: Added two disk writes to Metastor disks … writing 4 DDN disks, reading two metastor disks, reading four nighthawk2 gige's, writing four winterhawk gige's. Got 841MB/s with 85% system time, 8.2 % wait time.

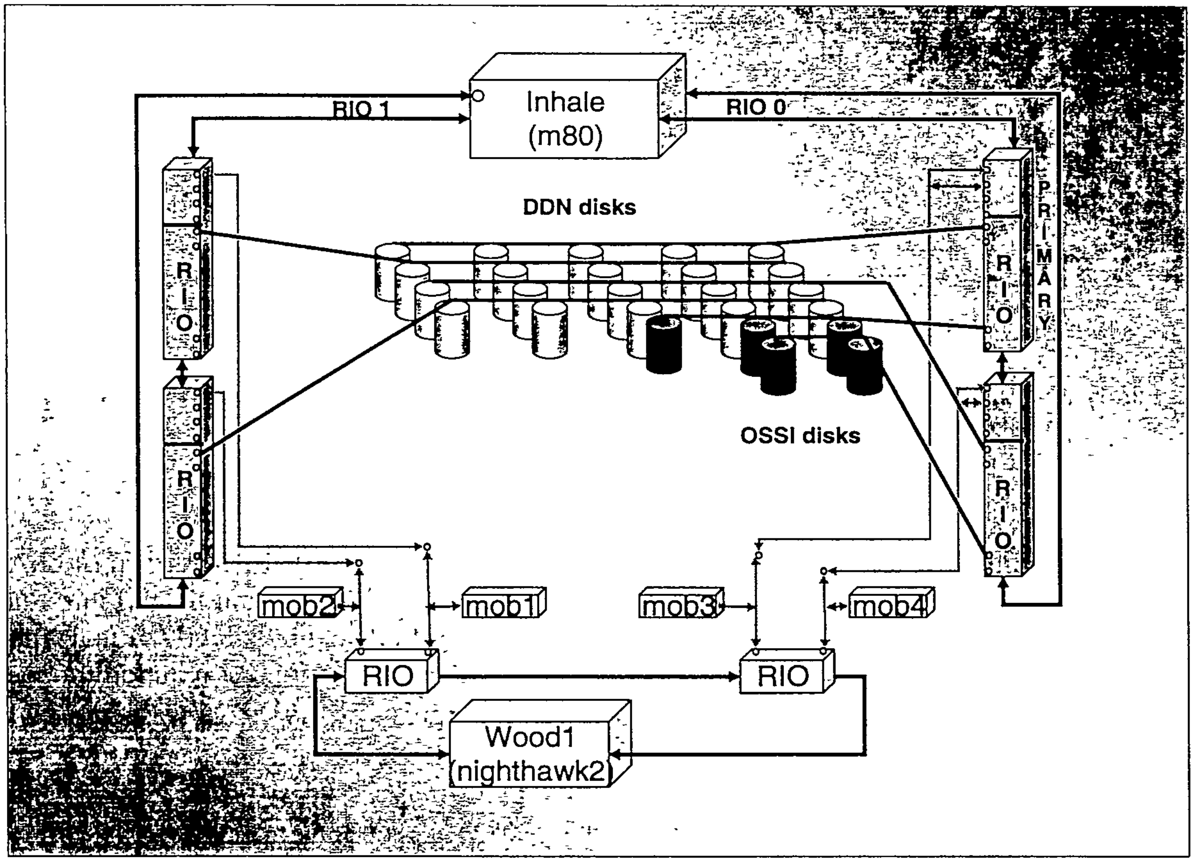
Interface	read	write
lpfc0	45	63
lpfc1	45	63
lpfc2	-	75
lpfc3	-	74
en2	60	59
en3	60	59
en4	60	59
en5	60	59
	330	511 = 841

Test4: Doubled the number of gigabit Ethernet data streams in each direction (8 each way). Got 934MB/s with 100% system time. I/O was split 358MB/s disk, 756MB/s network .

Interface	read	write
lpfc0	44	62
lpfc1	44	62
lpfc2	-	73
lpfc3	-	73
en2	70	81
en3	53	88
en4	70	81
en5	51	82
	332	602 = 934

The fact that the data rate increased shows that the I/O hub and 6XX bus is not yet saturated. The fact that we are out of CPU power (without adding much useful bandwidth) shows that we've saturated the machine and hit the point on the curve where performance drops dramatically. None of our interfaces are running at full speed.

11. Add four more cards (two gige NIC's and two emulex LP9000 HBA's) and again attempt to saturate the I/O hub and 6XX bus. In the new configuration there's one physical RAID disk on each of the HBA's. I had to split two of the jumbo subnets to accommodate the additional gige interfaces.



Test1: write four DDN disks, read 4 metastor, write two nighthawk nodes, write four winterhawk nodes. Got 1007MB/s with 67% system time and 22% wait time. This test achieved the highest aggregate data rates I got during testing. Even tho the RIO load is unbalanced (668MB/s on RIO 1 compared to 339MB/s on RIO 2) .. and most of the load was generated by writes (844MB/s write 163MB/s read) all the devices are running close to full speed. This test leads me to believe we can possibly saturate the I/O capability of the machine using 6 M80 processors rather than 8.

Interface	read	write
lpfc0	-	72
lpfc1	82	-
lpfc2	-	71
lpfc3	81	-
lpfc4	-	74
lpfc5	-	74
en2	-	90
en3	-	91
en4	-	90
en5	-	91
en6	-	96
en7	-	95
	163	844=1007

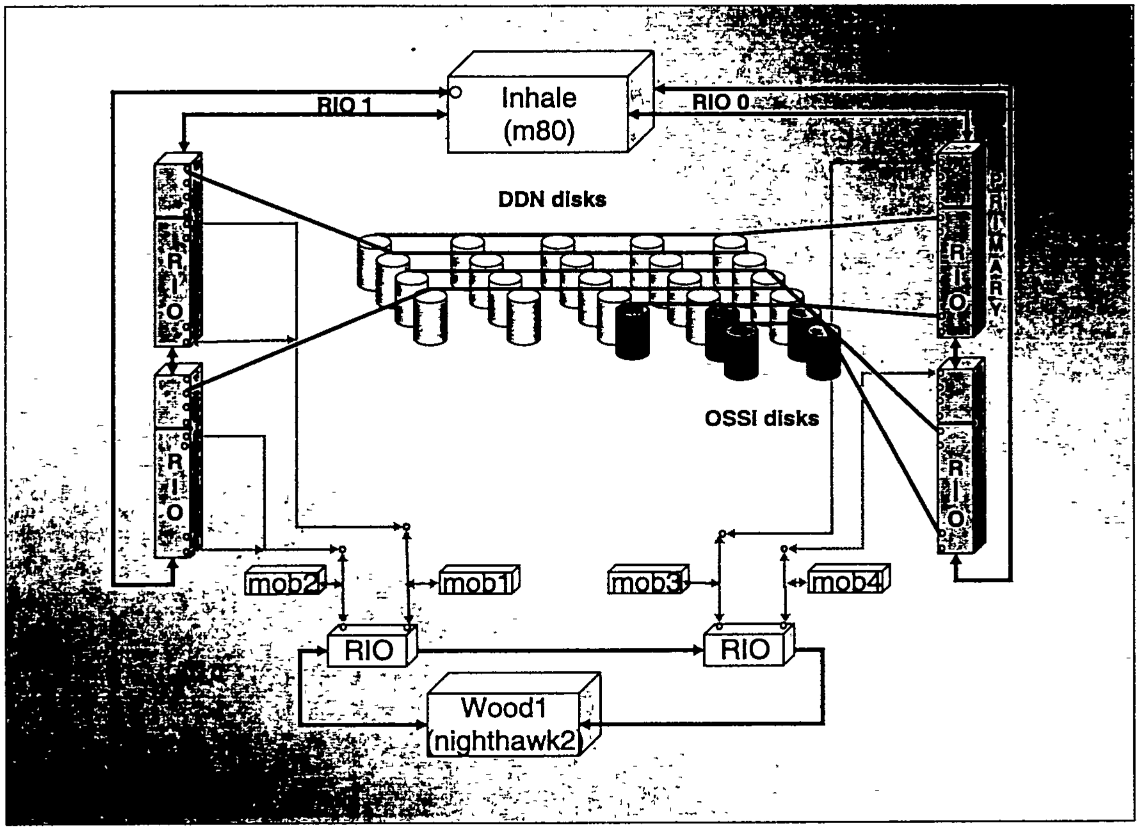
I modified the previous test slightly by adding four gige reads (one on each gige interface) originating at the nighthawk node (wood1). I achieved an aggregate data rate of 1038MB/s but the entire machine was saturated (99.9% system time).

Test2: read 6 disks while writing 6 gige interfaces. Got 887MB/s with 69% system time and 21% wait time. However this is a VERY significant test. The only change from the previous test is that four of the disk devices are reading rather than writing! Note that lpfc4-5 and en4-5 are running at considerably higher data rates than the other interfaces. The four interfaces that are running at the higher speeds were on the SAME RIO loop. The other eight interfaces were all on the same RIO drawer. In this configuration all the devices on either RIO loop can run concurrently if the other loop is idle. But they won't run at full speed together! This is an indication of a bandwidth limitation in the I/O hub or 6XX bus! This test doesn't seem make any sense at all. The I/O is more evenly distributed across the RIO buses and the read to write ratio is more evenly distributed yet performance has dropped off and one of the RIO loops appears to be saturated. Either of the RIO loops can run at full speed individually but they can't run together.

Interface	read	write
lpfc0	75	-
lpfc1	80	-
lpfc2	77	-
lpfc3	81	-

lpfc4	96	-
lpfc5	96	-
en2	-	53
en3	-	53
en4	-	86
en5	-	86
en6	-	52
en7	-	52
	505	382=887

12. Distribute the I/O load evenly across the RIO loops and retest.



In this configuration we have three PCI cards in each RIO drawer with one card in each PCI bus (two cards on a single bus but separate bus repeaters).

Test1: Host read simulation. Read all four DDN disks (97MB/s each possible), read two metastor's (82MB/s possible) while concurrently writing to three winterhawk gige's (en3-5, one winterhawk was down at this point), and three gige interfaces on the nighthawk node (en2,en6,en7).
We achieved 902MB/s with 76% system load and 16% wait.

Interface	read	write
lpfc0	81	-
lpfc1	88	-
lpfc2	80	-
lpfc3	81	-
lpfc4	88	-
lpfc5	81	-
en2	-	64
en3	-	63
en4	-	70
en5	-	70
en6	-	70

en7	-	66
	499	403=902MB/s

Test2: Host write simulation. The six disks are all capable of writing around 77MB/s each concurrently and the gige interfaces can read at around 82MB/s each when tested without the disk load. En3,4,5 reading from winterhawks, en2,6,7 reading from nighthawk node. Achieved 893MB/s with 84% system time 16% wait.

Interface	read	write
lpfc0	-	68
lpfc1	-	70
lpfc2	-	80
lpfc3	-	68
lpfc4	-	69
lpfc5	-	79
en2	77	-
en3	77	-
en4	76	-
en5	78	-
en6	75	-
en7	76	-
	459	434=893MB/s

```
Test3: Intermixed read/write simulation.
   Note here that the disks are all running at near their full
   bandwidths ... DDN's write at 78MB/s read at 96MB/s, metastor's read
   and write around 80MB/s.

   Achieved 987MB/s  ... using around 84% system and 16% wait.

      M80-device M80-rio-loop  m80-operation      data-source/sink
        en2          1           write                wood1-en2   ***
        en3          1           read                 mob2-en2
        en4          2           write                mob3-en2
        en5          2           read                 mob4-en2
        en6          2           write                wood1-en3   ***
        en7          2           read                 wood1-en5
      lpfc0          1           write            hdisk2   ddn_t1_test
      lpfc1          1           read             hdisk7   meta_hpss_1
      lpfc2          1           write            hdisk9   ddn_t2_test
      lpfc3          1           read             hdisk14  meta_hpss_0
      lpfc4          2           write            hdisk18  ddn_t3_test
      lpfc5          2           read             hdisk23  ddn_t4_test
```

*** These interfaces are on the same nighthawk RIO drawer

Interface	read	write
lpfc0	-	72
lpfc1	82	-
lpfc2	-	69
lpfc3	81	-
lpfc4	-	72
lpfc5	94	-
en2	-	87
en3	84	-
en4	-	99
en5	83	-
en6	-	86
en7	78	-
	502	485=987MB/s

13. At this point we are able to saturate the I/O capability of the system. We want to find out how much CPU power is required to saturate the system's I/O capabilities. Using the system's service processor capabilities we disable one cpu at a time and rerun our tests until aggregate I/O throughput degrades.

    Early tests show that if we add additional gige streams, we saturate the CPU capability of the machine. However in the optimal configuration tested above with six gige streams there's CPU left. Adding additional I/O streams (which requires more CPU) does not improve the overall I/O throughput significantly. Individual gigabit Ethernet tests show that it requires almost a dedicated CPU to maintain a full bandwidth transfer. The I/O capability of the machine can be saturated by the 8 CPU's as demonstrated by the fact that all I/O streams are degraded equally in the preceding tests. Note that even with 8 processors per M80 the ratio of I/O capability-to-processor is significantly lower than with the winterhawks (see following winterhawk test results). With the winterhawks we achieved 280MB/s with 4 processors. With the M80 we can achieve 980MB/s with 8 processors (70MB/s per processor on winterhawk compared to 122MB/s per processor with the M80). However, since the M80 can be purchased with 6 CPU's (one four processor module and one two processor module) we could consider a six processor configuration if the savings are significant.
    (SHOULD TEST with 6 CPU's didn't realize we could get a 2 CPU module until recently) ....

HPSS Based Testing (Jim Daveler)

At this point we will have a pretty good idea of the I/O capabilities of the current test configuration. We will configure the system as an HPSS mover and verify system behavior running under the actual storage application.

Since we do not have sufficient tape bandwidth in the I/O testbed to stress a system of this capability, we will simulate the HPSS migration load through host read operations. Since the same path (disk device to/from gige interface) will be used for tape access a host read or write will provide a valid simulation.

The preceding simulations have shown that the hardware should be able to support six concurrent (nearly) full speed data streams. Our challenge is to configure the HPSS system in a fashion that causes all the interfaces will be used. Our goal is to be able to exploit the parallelism inherent in HPSS without undue administrative complication. To the best of my knowledge, no HPSS site has configured a system running only movers on a platform with the power of the M80 and having the network bandwidth provided by our I/O testbed. A complete HPSS system was configured on an IBM S80 at ORNL but they did not have as many I/O cards or a robust multi-subnet gigabit Ethernet infrastructure.
An HPSS mover facilitates data transfers between storage clients, other HPSS movers and devices. We must therefore insure that both the clients and other HPSS movers spread their data transfers across all six gigabit interfaces available on the M80. To accomplish this goal, we configured six HPSS movers on the M80 and set the HPSS "data-host" address for each mover to a different M80 gigabit Ethernet interface. This configuration insures that when another HPSS mover transmits data to the M80 the M80 interface used will be determined by the destination data mover process. Because our current production HPSS movers all have their "data host" addresses set to their SP switch IP address we must assign hard routes on the M80 to handle the situation. The hard route on the M80 will be assigned to the production mover's jumbo gigabit Ethernet address. Because the existing HPSS data movers are distributed across the center's four jumbo subnets, this assignment will naturally distribute the data flow between the M80 and the existing HPSS data movers across at least four of the M80's gigabit interfaces. Data movement between the storage clients and the data mover processes is parallelized by the HPSS client logic which allows a storage client to specify which client interfaces are available for a transfer. This logic will distribute the load across the specified interfaces. Because the M80 will have six interfaces and we currently have only four jumbo gige subnets, we will have to split two of the jumbo subnets on the M80 side to accommodate the additional interfaces. Use of IP aliases and thoughtful initial assignment of IP addresses can help to distribute the load across six jumbo subnets rather than four. Until our data center increases the number of gigabit Ethernet subnets, two of the M80 gigabit Ethernet interfaces are likely to be underutilized and care should be exercised in the HPSS configuration to insure that two HPPS data stripes are not

assigned to the same jumbo subnet. An easy way to avoid problems is to establish an HPSS administrative convention that stripes are assigned only to the first four movers on an M80.

While running these tests we also discovered that AIX can not read a file system at much more than 200MB/s (even when the data is generated in memory – reading from /dev/zero or a sparse file!).

1. Configure HPSS devices.

   Because the LLNL production environment currently supports four jumbo gigabit interfaces, the optimum (widest possible) stripe width is 4. We configured a four-wide striped disk on M80 movers 1-4 and two single-wide disks on mover5 and mover6. With this configuration we should be able to demonstrate individual disk transfer rates, striped transfer rate, and aggregate (mover saturated) transfer rates. The four-wide stripe was configured on the DDN disk and the individual HPSS volumes were configured on the two metastor disks. We originally configured virtual volume sizes of 32MB but reconfigured the four way striped COS and one un-striped COS with an 8MB VV (virtual volume) size in an attempt to increase performance. PFTP performance was not affected significantly by the VV size however HSI performance improved. Network and operating system parameters used during the HPSS simulation testing were used for this test. In the following tests, the HPSS core servers were running on one winterhawk node and the HPSS database (Transarc SFS) was running on another winterhawk nodes. I suspect that the write rates suffered due to the fact SFS was not running on the same physical hardware. Other HPSS testing has not shown severe discrepancies between read and write rates.

- Disk Storage Class List -

SC ID	Name	Xfer Media Type	SSeg Rate (kB/s)	Est Size min/max	Avg # SSegs	PV Size	Media VVBS	Str Wid	Block Size	Thresh (% use) Wrn/Crt	Mig Pol ID	Prg Pol ID
1111	OSSI	Default	50000	8MB/ 64MB	4	49GB	8MB	1	512K	80/ 90	0	0
	1-Way Disk(sc=1111)											
2222	OSSI 1	Default	50000	128MB/ 1GB	4	49GB	32MB	1	512K	80/ 90	0	0
	1-Way Disk(sc=2222)											
4000	DDN	Default	200000	128MB/128MB	4	49984MB	8MB	4	512K	92/ 95	0	0
	4-Wide Disk(sc=4000)											

```
Client Side settings(wood1)
       PF
       PFTP Client Interfaces = {
          wood1 wood1.llnl.gov = {
                Default = {
                    TP Client = {
           Default COS = 4000
           Protocol = PDATA_ONLY
           Socket Buffer Size = 4MB
           Parallel Block Size = 8MB
```

```
        Transfer Buffer Size = 8MB

        MAX Ptran Size = 250GB
    }134.9.33.133
                134.9.33.149
                134.9.33.165
                134.9.33.181
            }
        }

    }

    Network Options = {
       Default = {
            Default = {
                NetMask = 255.255.255.255
                RFC1323 = 1
                SendSpace = 4MB
                RecvSpace = 4MB
                WriteSize = 32KB
                TcpNoDelay = 1
            }
       }
    }
```

2. Test data rate to an individual HPSS volume.

   We tested individual data rates by reading and writing one of the un-striped classes of service (2222) from the mob4 "winterhawk" system.

Operation	Data Rate (MB/s)
read	75
write	41

3. Test concurrent data rates to two un-striped classes of service.

Operation	mob4 (MB/s)	mob3 (MB/s)	Total (MB/s)
read	72	72	144
write	40	40	80

4. Test data rate to a striped HPSS volume.

   We read and wrote the 4 way class of service from the nighthawk system.

Operation	Data Rate (MB/s)
read	215
write	140

Test aggregate (all six movers active) data rate.

Operation	mob4 (MB/s)	mob3 (MB/s)	wood1 (MB/s)	Total (MB/s)
read	70	70	203	343
write	40	40	142	222

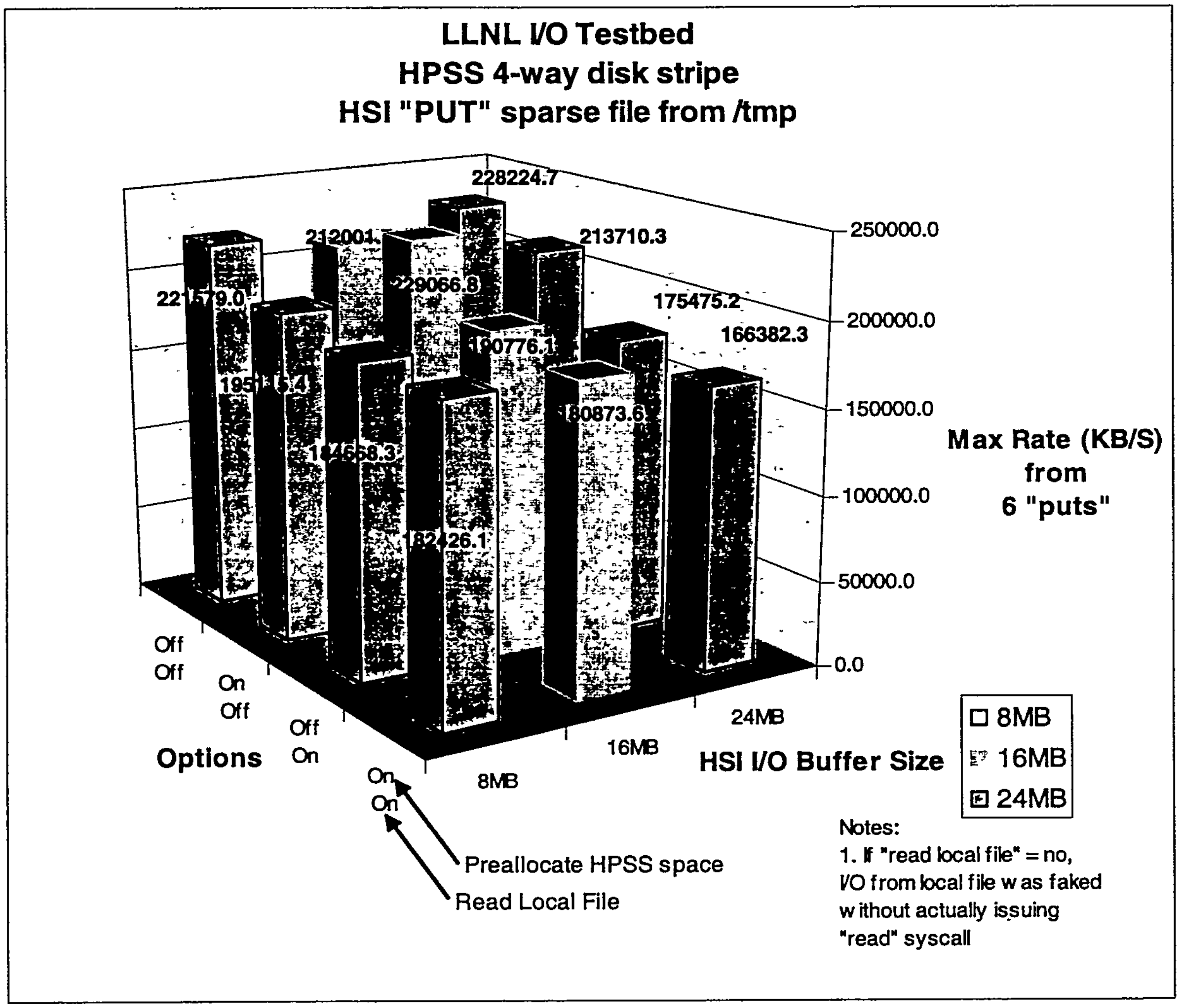
HSI data rates (Mike Gleicher)

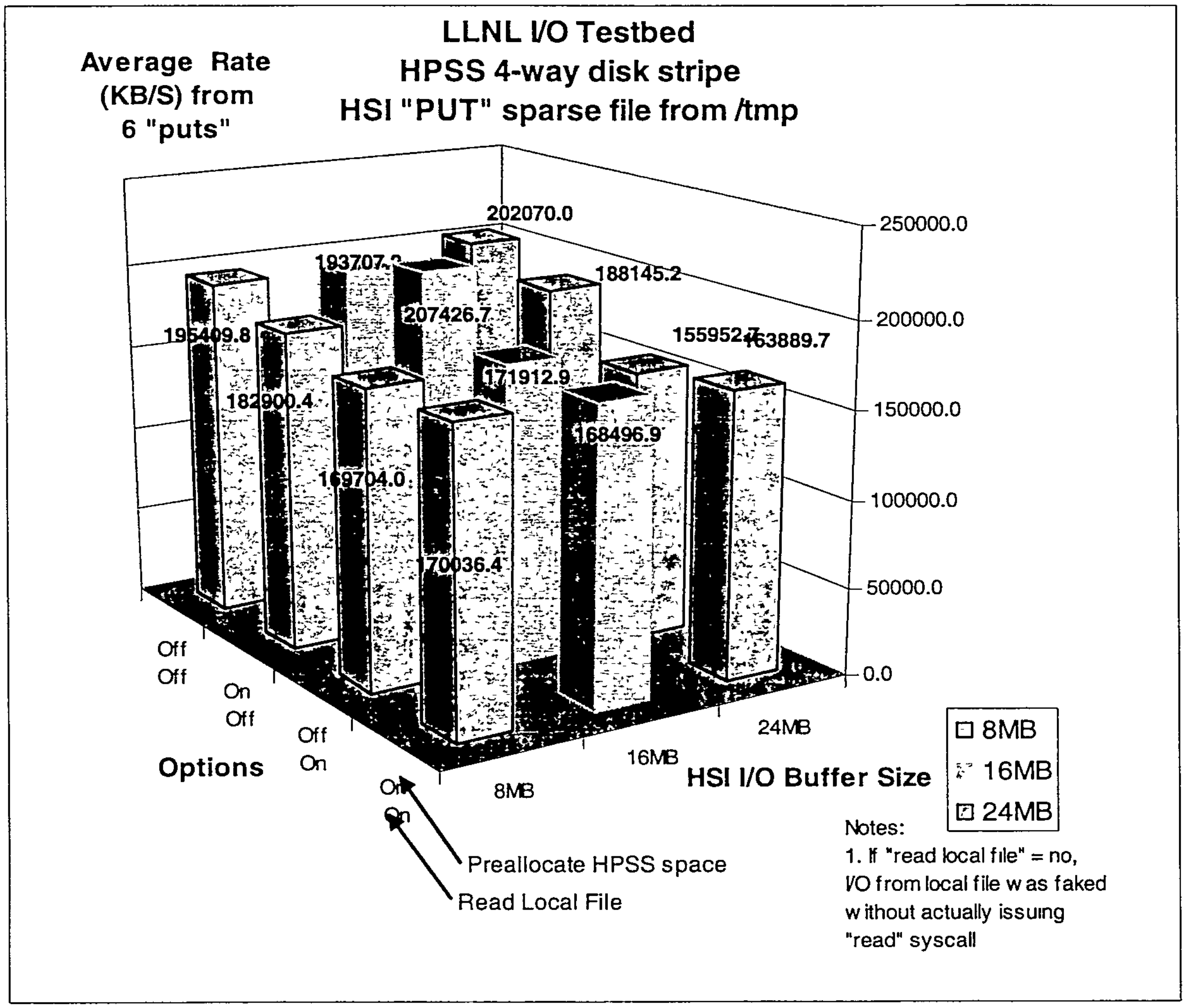
```
From: Mike Gleicher <gleicher@toofast10.llnl.gov>
     To: jdaveler@toofast10.llnl.gov, kfitz@llnl.gov, mkg@san.rr.com
     Subject: wood1 - slightly better results

     Jim/Keith -

     Here are a couple of tests I just ran from wood1 -
     ---------------------------------------------------------
     M:[nmob2]/home/jdaveler->get /dev/null : sparse.hoho
     get  /dev/null : /home/jdaveler/sparse.hoho (2002/02/13 17:15:38
            2147483647 bytes, 223554.9 KBS )
     M:[nmob2]/home/jdaveler->get /dev/null : sparse.hoho
     get  /dev/null : /home/jdaveler/sparse.hoho (2002/02/13 17:15:38
            2147483647 bytes, 303720.6 KBS )

     Date: Wed, 13 Feb 2002 19:59:23 -0800
```

LLNL Testbed
HSI on wood1, 4 jumbo-frame gige adaptors
4 way disk stripe on (??) - 4 jumbo frame gige adaptors
16MB mover buffer

I/O Buffer Size	Local Reads	Preallocate (on/off)	MAX	MIN	Average	Median
8MB	Off	Off	221579.0	174485.7	195409.8	189089.5
8MB	Off	On	195115.4	159259.4	182900.4	186584.0
8MB	On	Off	184668.3	150111.2	169704.0	168788.8
8MB	On	On	182426.1	161112.7	170036.4	169887.9
16MB	Off	Off	212001.7	164989.7	193707.2	196157.8
16MB	Off	On	229066.8	187135.0	207426.7	205076.8
16MB	On	Off	190776.1	156641.7	171912.9	170458.0
16MB	On	On	180873.6	156067.4	168496.9	168478.2
24MB	Off	Off	228224.7	166164.0	201116.8	202070.0
24MB	Off	On	213710.3	174740.1	191644.9	188145.2
24MB	On	Off	175475.2	133465.4	157693.2	155952.7
24MB	On	On	166382.3	130454.1	158582.7	163889.7

I/O Buffer Size	Local Reads	preallocate(on/off)	Megabytes per Second					
			Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6
8MB	Off	Off	179091.8	221579.0	174485.7	219123.1	196702.5	181476.4
8MB	Off	On	193788.2	176071.1	195115.4	159259.4	186121.5	187046.5
8MB	On	Off	184668.3	173540.9	150111.2	182239.1	163627.5	164036.7
8MB	On	On	172726.8	173921.8	167049.0	161112.7	182426.1	162982.1
16MB	Off	Off	164989.7	183414.1	183672.1	209521.9	208643.5	212001.7
16MB	Off	On	187135.0	225996.6	201986.4	208167.1	229066.8	192208.0
16MB	On	Off	169337.0	164550.3	156641.7	190776.1	171579.0	178593.0
16MB	On	On	156067.4	163035.1	174048.7	168345.3	168611.0	180873.6
24MB	Off	Off	166164.0	214557.9	193614.4	228224.7	210500.3	193639.7
24MB	Off	On	174740.1	194655.7	213710.3	210383.0	181634.6	174745.4
24MB	On	Off	159048.4	133465.4	152462.4	172850.6	152856.9	175475.2
24MB	On	On	166382.3	163538.1	162526.2	130454.1	164354.1	164241.2


LLNL I/O Testbed
HPSS 4-way disk stripe
HSI "PUT" sparse file from /tmp
228224.7
212001.7
213710.3
221579.0
229066.8
175475.2
166382.3
195115.4
190776.1
180873.6
174668.3
182426.1
250000.0
200000.0
150000.0
100000.0
50000.0
0.0
Max Rate (KB/S) from 6 "puts"
Off Off
On Off
Off On
On On
Options
8MB
16MB
24MB
HSI I/O Buffer Size
8MB
16MB
24MB
Preallocate HPSS space
Read Local File
Notes:
1. If "read local file" = no, I/O from local file was faked without actually issuing "read" syscall



LLNL I/O Testbed
HPSS 4-way disk stripe
HSI "PUT" sparse file from /tmp
Average Rate
(KB/S) from
6 "puts"
202070.0
193707.2
188145.2
195409.8
207426.7
155952.7
163889.7
182900.4
171912.9
168496.9
169704.0
170036.4
250000.0
200000.0
150000.0
100000.0
50000.0
0.0
Off
Off
On
Off
Off
On
On
On
Options
8MB
16MB
24MB
HSI I/O Buffer Size
8MB
16MB
24MB
Preallocate HPSS space
Read Local File
Notes:
1. If "read local file" = no,
I/O from local file was faked
without actually issuing
"read" syscall

Six Processor testing

Processors are expensive (~ 25K each)! HPSS simulations have shown that the I/O bandwidth can be saturated at around 75% cpu utilization. In this test I went into the service processor menu and turned off the last two processors (6 and 7). I then reran the HPSS simulation tests.

1. HPSS read simulation: achieved a total of 874MB/s

Interface	read	write
lpfc0	83	-
lpfc1	80	-
lpfc2	83	-
lpfc3	81	-
lpfc4	90	-
lpfc5	90	-
en2	-	70
en3	-	75
en4	-	74
en5	-	74
en6	-	74
	507	367

2. HPSS write simulation: achieved a total of 726MB/s

Interface	read	write
lpfc0	-	64
lpfc1	-	66
lpfc2	-	39
lpfc3	-	65
lpfc4	-	78
lpfc5	-	79
en2	68	-
en3	67	-
en4	67	-
en5	67	-
en6	60	-
en7	60	-
	335	391

SP Attached M80 Analysis (Todd Heer)

Certain functions must be present to enable a machine to be managed and administered like the storage groups current SP nodes. This requirement gives continuity of administration for new machines, ensuring that they can benefit from the commands, tools, and functionality already present. We must be mindful to not linearly add administrative overhead for each new machine introduced.

This functionality is given when the M80 is attached to the control workstation via a cable which interfaces to the M80 server via the *PCI Card for SP Control Workstation Attachment* (feature #3154). Internally, this pci card is placed in slot 7 of the primary I/O drawer of the M80 and a cable is run from the card to a specific connector on the planar board. The test was performed on AIX 433 maintenance level 8 with PSSP version 3.2, including the requisite APAR IY16350.

Here is what I found:

1) *Can the M80 be powered off and on using SP utilities?*
   Yes. Much like any normal (in-frame) SP node, the hardmon daemon does control power to the M80 through tools such as **spmon**. This was verified.

2) *Does the SP console (s1term) perform properly?*
   Yes. This is often accessed by executing **spmon –open node<X>**. This is where we had some issues initially, however once the console was changed to /dev/tty0 on the M80 it worked fine. This likely would not have been an issue if we had installed the M80 from the cws directly initially (rather than retrofitting it to the SP – which is a good way to learn how things work).

3) *Does SP software run correctly on the M80 as an integrated environment?*
   Yes, as far as we tested. I was able to get host responds up, which means the High Availability Infrastructure was operating normally. There was no high performance switch connection, so switch responds was not tested.

4) *Did Kerberos authentication work?*
   N/A. This was not tested. I surmise, based on what I did test, that indeed it would work as designed. Again, we retrofitted an already install standalone M80 into an existing SP environment. Had we successfully installed from scratch, the proper Kerberos files would have been sent to the client (M80) from the cws.

5) *Does a network install perform properly?*
   This can be answered in two parts. First, there are the mechanics of the install. Those being a network boot, successful transmission via bootp and tftp of a boot image, nim install beginning, mksysb image being transferred over the network, and having it installed to disk on the client. This part worked. I did encounter an issue which went unresolved for the last part of a successful node install, the

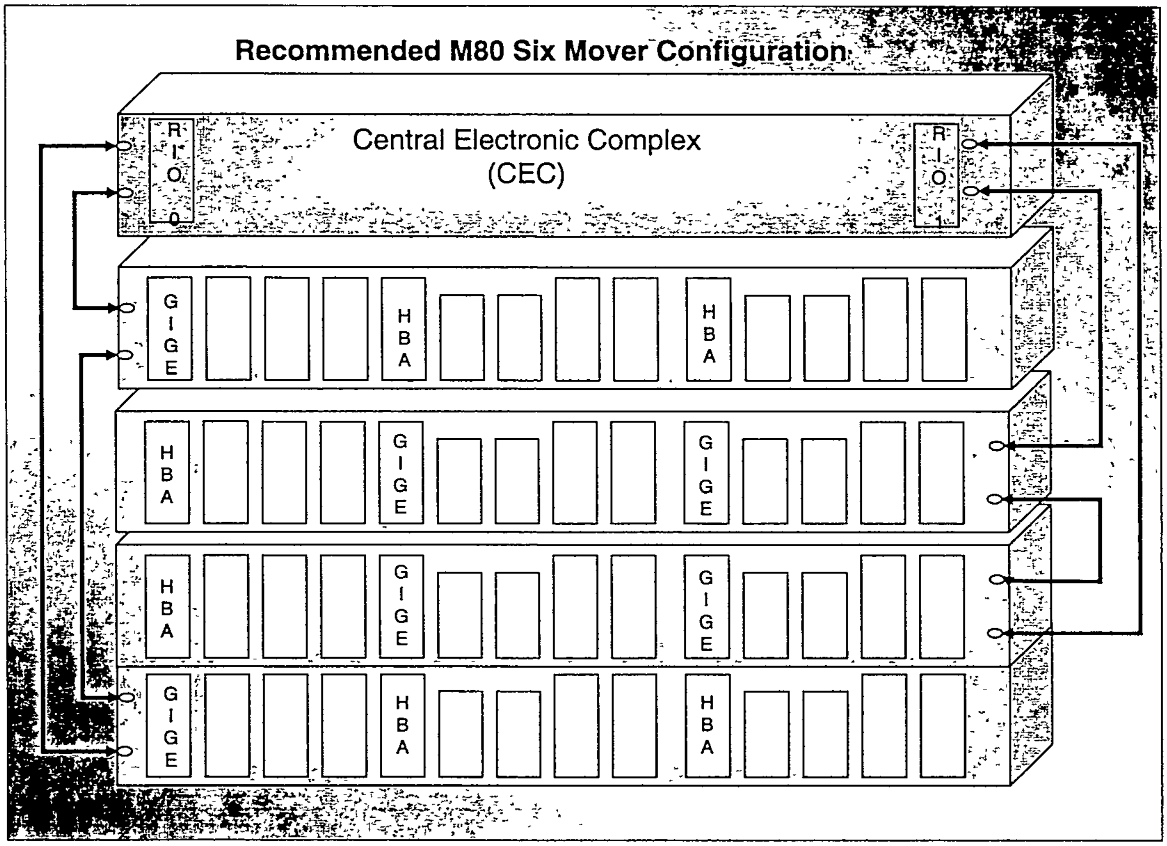
customize portion. Given time restrictions, I was not able to determine why the customize hung at a led, but having debugged many SP node customizations, I can safely say that this would not be a problem. I assume there was a permissions problem with a config file or perhaps a subset of a network problem. The important thing is that for the purposes of a network install, the machine behaved exactly like a normal SP node. In that sense, the install did perform properly.

Oustanding Issues

Having only one of these systems on hand to test, we can't verify the scalability of two or more systems. Presumably we would add systems over time, as we augment HPSS movers and our throughput needs increase. What we don't know is the effect of several of these machines on a control workstation and what the limit is. This question would have to be answered within some acceptable margin of error before choosing to go down this path. IBM development/support could be consulted on this issue.

Todd Heer
Data Storage Group
LLNL
theer@llnl.gov

Conclusions



1. You can achieve a data rate of around 320MB/s on the 64bit 66MHz PCI bus which can run at a theoretical rate of 528MB/s (around 60% of theoretical). This translates to the fact that you should install no more than two full duplex 100MB/s PCI cards (like IBM gige's or emulex HBA's) in each PCI bus.

2. You can achieve a data rate of around 375MB/s on a RIO connection which has a 500MB/s theoretical transfer rate. (around 75% theoretical). This means that either of the two PCI buses could nearly saturate a RIO connection if you installed four cards in the PCI bus. However the RIO speed isn't a limitation if you adhere to rule 1 … only two cards in a PCI bus. This means only four cards in a RIO drawer in which case there's enough RIO bandwidth (375MB/s in each direction) to handle all four cards.

3. An individual RIO drawer is bound to a single RIO connection on the I/O hub card. This limits an individual RIO drawer to 375MB/s in each direction. This means that you must do a mixture of reads and writes in order to utilize the total bandwidth offered by the two PCI buses in a RIO drawer.

4. A pair of RIO drawers have a potential data rate of 750MB/s (375MB/s in each direction) if it was configured with 2 cards per bus and therefore four cards per RIO drawer.

5. Four fully loaded RIO drawers could provide an aggregate bandwidth of 1500MB/s BUT ... the I/O hub module or 6XX memory bus saturates at around 1000MB/s.

6. In order for data to "stream" through the M80 the data must traverse the I/O subsystem twice. Once as the data is written into the M80 memory and once as the data is read out of the M80 memory. To move 100MB/s of data through a mover requires 200MB/s of mover bandwidth. It seems reasonable to assume that each data stream through the mover can achieve a data rate of 80MB/s (when we get the emulex HBA's fixed). Assuming that the M80 is capable of handling 1000MB/s that comes out to a bit over 6 concurrent data streams. Six streams implies 12 cards. However since each card is capable of full duplex operation 12 cards effectively over allocates the M80 by a factor of two.

7. If you assume that you only get 60MB/s per stream, you can configure 8 movers (60MB/s * 8 streams =480MB/s) on the M80. You actually require 2* 480MB/s because data must traverse both the NIC and HBA for a total of 960MB/s.

8. I was not able to read an AIX file system at much more than 200MB/s even when reading from /dev/zero or a sparse file. I also suspect that you can not write to GPFS from an individual node at much over that rate. I could not test this theory on the I/O testbed Nighthawk system because the SSA based GPFS system was only designed to deliver 100MB/s. However, if this is theory is valid it doesn't make sense to stripe more than three disks capable of running at 80MB/s because the client can not deliver the data fast enough.

Appendix A: Network options

```
</>no -a
         extendednetstats = 0
                  thewall = 1048496
               sockthresh = 85
                   sb_max = 2097152
                somaxconn = 1024
      clean_partial_conns = 0
        net_malloc_police = 0
                  rto_low = 1
                 rto_high = 64
                rto_limit = 7
               rto_length = 13
          inet_stack_size = 16
              arptab_bsiz = 7
                arptab_nb = 25
               tcp_ndebug = 100
                   ifsize = 35
                 arpqsize = 5
                 ndpqsize = 50
             route_expire = 5
       send_file_duration = 300
                 fasttimo = 200
          routerevalidate = 0
                nbc_limit = 786352
            nbc_max_cache = 131072
            nbc_min_cache = 1
                 nbc_pseg = 0
           nbc_pseg_limit = 16777192
                 strmsgsz = 0
                 strctlsz = 1024
                 nstrpush = 8
                strthresh = 85
                psetimers = 20
              psebufcalls = 20
               strturncnt = 15
             pseintrstack = 12288
                lowthresh = 90
                medthresh = 95
                 psecache = 1
          subnetsarelocal = 1
                   maxttl = 255
                ipfragttl = 60
          ipsendredirects = 1
             ipforwarding = 0
                  udp_ttl = 30
                  tcp_ttl = 60
               arpt_killc = 20
            tcp_sendspace = 655360
            tcp_recvspace = 655360
            udp_sendspace = 655360
            udp_recvspace = 655360
           rfc1122addrchk = 0
           nonlocsrcroute = 1
            tcp_keepintvl = 150
```

```
             tcp_keepidle = 14400
                bcastping = 0
                 udpcksum = 1
             tcp_mssdflt = 9000
          icmpaddressmask = 0
             tcp_keepinit = 150
ie5_old_multicast_mapping = 0
                  rfc1323 = 1
         pmtu_default_age = 10
 pmtu_rediscover_interval = 30
        udp_pmtu_discover = 1
        tcp_pmtu_discover = 1
                ipqmaxlen = 128
       directed_broadcast = 1
        ipignoreredirects = 0
           ipsrcroutesend = 1
           ipsrcrouterecv = 0
        ipsrcrouteforward = 1
       ip6srcrouteforward = 1
               ip6_defttl = 64
                ndpt_keep = 120
           ndpt_reachable = 30
             ndpt_retrans = 1
               ndpt_probe = 5
                ndpt_down = 3
            ndp_umaxtries = 3
            ndp_mmaxtries = 3
                ip6_prune = 2
            ip6forwarding = 0
              multi_homed = 1
                 main_if6 = 0
               main_site6 = 0
              site6_index = 0
                 maxnip6q = 20
          llsleep_timeout = 3
              tcp_timewait = 1
         tcp_ephemeral_low = 32768
        tcp_ephemeral_high = 65535
         udp_ephemeral_low = 32768
        udp_ephemeral_high = 65535
                  delayack = 0
             delayackports = {}
                      sack = 0
                 use_isno = 1
               tcp_newreno = 0
           tcp_nagle_limit = 65535
</>
```

Appendix B: Outstanding Issues

1. For some reason, the cisco 6509 disables the 100Mb/s Ethernet interface built into the primary RIO drawer when the system goes down. "interface show status" on the cisco shows the link in err-disable state.

   Setting the following options on the csico seems to mitigate the problem:
   "errdisable recovery cause link-flap"
   "errdisable recovery interval 30"

2. The emulex LP8000 and LP9000 adapters lockup occasionally when volume groups are being varied online. Using the emulex diagnostic "lputil" to reset the adapter unhangs the adapter (most of the time).

   We are working with emulex to resolve the issue.

3. When you install multiple HBA's on a single PCI bus using the M80 hot swap logic and then run "cfgmgr" to detect and configure the devices only the first HBA on the channel is detected. The other HBA(s) give "bus conflict" error messages if you examine the verbose "cfgmgr" output.

   Running "cfgmgr –p1" gets around the problem. Emulex will fix the problem in a later release.

4. The LP8000 and LP9000 adapters do not write as fast as on the IBM "winterhawk2" nodes. We can achieve write rates of 96MB/s on winterhawk nodes we only get around 80MB/s from the M80. System, driver, and HBA microcode levels are the same.

   We are working with emulex to resolve the issue. It would be interesting to test the IBM supported version of the emulex LP9000.

5. Concurrent read/write rates on the emulex LP8000 and LP9000 adapters are low. We only achieved 112MB/s on concurrent read/writes on the M80. We can achieve 150-160MB/s on the winterhawk2 nodes and this rate may be limited by the 64bit 33MHz PCI bus which only has a theoretical data rate of 264MB/s.

   We are working with emulex to resolve the issue. Testing the IBM version of the LP9000 would be interesting.

6. TCP/IP over fibre channel using LP8000 or LP9000 can only write at around 15MB/s. Suspect the problem is related to the low write rates described in the previous two items.

Appendix C: Individual Card performance (reference)

(all values in MB/s)

IBM gigabit Ethernet NIC Performance (MB/s)

Architecture	read	write	read/write
IBM winterhawk2	105	100+	160
IBM nighthawk2 *	102	104	160
IBM M80	109	105	170

* if you put multiple gige NIC's in a nighthawk RIO drawer, read rate for two NICs in a RIO (different buses) degrades to 82MB/s. Write rate degrades to 88MB/s and aggregate degrades to 116MB/s.

Emulex LP8000 HBA Performance (MB/s)

Architecture	read	write	read/write
IBM winterhawk2	98	98	160
IBM nighthawk2*	?	?	?
IBM M80	98	77**	112**

* if you put multiple HBA's in a nighthawk RIO drawer, data rates in a RIO drawer are limited by the RIO connection .... 250MB/s theoretical

** I suspect the problem with data rates is the emulex adapter NOT the M80!

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.